Towards Developing Multi-Personal-Node Applications

Han Feng[1] Yasushi Shinjo[1]

1. Introduction

A single user often simultaneously uses multiple computing nodes, such as a PC and smartphone. We call such nodes *personal nodes*. A user can also run a same name application on multiple personal nodes at the same time. For example, a user can run a web browser or instant messenger on a PC and a smartphone. In this case, the user expects to switch and operate in these two personal nodes and expects the application running on them to have the ability to share resources and data. Some applications, such as popular web browsers and the messenger application for iOS and macOS provide limited support for sharing by exchanging states through centralized servers. However, they can not truly achieve seamless user experience and resource sharing among multiple personal nodes. Furthermore, coordination through central servers can lead to privacy violations.

In this research, we develop one application that works across multiple personal nodes. We call such applications *multi-personal-node applications* (MPN applications). When a user uses an MPN application with multiple personal nodes, the user feels as if the user is using one application. This way, users can take full advantage of multiple personal nodes and get a seamless user experience.

To support the development of MPN applications, we are implementing a framework. This framework provides the service discovery mechanism with multicasting and the distributed service invocation mechanism with Remote Procedure Calls (RPCs) and event queues. The ultimate goal of this framework is to let developers have the ability to develop MPN applications in the same way that they develop traditional cross-platform applications.

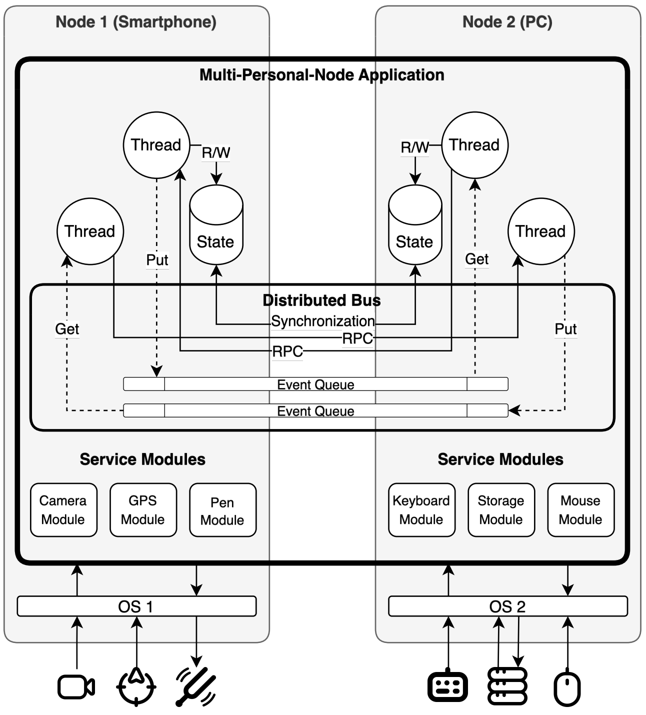

Fig. 1 A multi-personal-node application running on a smartphone and PC.

2. Design of the Framework for MPN Applications

2.1 Overall Architecture

Figure 1 shows the overall architecture of our framework for MPN applications. In this figure, a single MPN application runs on two personal nodes. Node 1 is a smartphone or tablet, with a camera, GPS, and vibrator, running an OS like Android or iOS. Node 2 is a PC with a keyboard, large storage, and mouse, running an OS like Windows or macOS. A thread runs on an OS and a node, and exchanges messages with threads on a local or remote node. The states are variables, typically bounded to a node, and accessed by local threads. The MPN application includes *a distributed bus*, which provides inter-node communication. We discuss this in detail in the following section.

In an MPN application, each node can contain service

[1] University of Tsukuba

modules to access the OS and hardware devices through native APIs. For example, an Android node contains the service modules to access a camera through the Java API and a macOS node contains the service modules to access a keyboard through the Objective-C API.

2.2 Distributed Bus

The distributed bus plays a similar role as the distributed kernel of a distributed operating system. The distributed bus runs on all personal nodes and provides inter-node communication with RPCs and event queues. Events are typically used through callback functions in many programming languages.

The distributed bus provides a discovery service of service modules. In each node, the distributed kernel maintains the list of available service modules. These service modules take charge of accessing specific local hardware devices. When an application thread tries to use a service module, it asks the distributed bus to discover the module. The distributed bus uses multicast DNS (mDNS) and publishes the request for the module to the LAN. When another distributed bus receives the request and the node has the module, the distributed bus answers the request. Finally, these two distributed kernels in the two nodes collaborate and make the module accessible with RPCs and event queues.

2.3 MPN Application Styles

We support the development of the following three styles of MPN applications in our framework.

(1) RPC style

In this style of application, one node is defined as the *core node* for each state. Threads running on the core node access the state directly. Threads running on another node access the state through RPCs and events. For this purpose, the core node runs the server thread of the RPC.

The advantage of this style is that it can be used without changing the logic of a traditional application running on a single node. Since a state resides in a single node, no update conflicts arise. The weakness of this style is that if the core node terminates or the network to the core node is disconnected, the application stops working.

(2) Mobile Agent Style

This style of application is implemented based on mobile agent technologies. For example, when a user goes out, the user can move all the states to the smartphone node and continue running the application.

The weakness of this style is that each node cannot continue execution independently when communication between these nodes is lost.

(3) State Replication Style

This style of an application replicates states to multiple nodes, allowing each to continue execution even when communication among nodes is lost. Because state replication is dependent on the semantics of applications, no general solution exists.

To help maintain consistency across replicas, we plan to provide Conflict-free Replicated Data Type (CRDT) support in the distributed bus. A CRDT is a data structure to grantee eventual consistency in distributed systems.

3. Related Work

Mobile Plus[1] and FLUID[2] focus on seamless interaction at the user experience level, and improve the consistency of the user interface by sharing the state of the environment across devices. Although they have similar goals to our research, they focus more on the migration at the UI level and only support the Android system. In this research, we aim to develop an MPN application that simultaneously uses different OSs such as Android and macOS.

4. Summary

In this research, we are implementing multi-personal-node applications and the framework for building such applications. Each multi-personal-node application consists of a distributed bus, threads, states, and service modules to access native services of OS and I/O devices. We are implementing MPN applications on an Android smartphone and a macOS PC. We are interested in converting conventional cross-platform applications into multi-personal-node applications.

References

[1] Sangeun Oh, Hyuck Yoo, Dae R Jeong, Duc Hoang Bui, and Insik Shin. Mobile plus: Multi-device mobile platform for cross-device functionality sharing. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 332–344, 2017.

[2] Sangeun Oh, Ahyeon Kim, Sunjae Lee, Kilho Lee, Dae R Jeong, Steven Y Ko, and Insik Shin. Fluid: Flexible user interface distribution for ubiquitous multi-device interaction. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.